

Trust Management: An Overview

Matt Blaze

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE 04 NOV 2009		2. REPORT TYPE		3. DATES COVERED 00-00-2009 to 00-00-2009	
4. TITLE AND SUBTITLE Trust Management: An Overview				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Pennsylvania, Computer and Information Science, Philadelphia, PA, 19104				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES ONR MURI Review, Nov 2009.					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 20	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

“Classic” Trust Management

- For answering questions of the form: “Should I perform this (dangerous) action?”
- Systematic approach to managing
 - security policies
 - credentials
 - trust relationships
- Term coined in 1996
 - Blaze, Feigenbaum, Lacy. “Decentralized Trust Management.” IEEE S&P (Oakland), 1996.

Trust Management: Compliance Checking

- Provides advice to applications on whether “dangerous” actions should be permitted
- Compliance checker uses local policy & signed credentials in making these decisions
 - guarantees that only actions that conform to policy will be approved
- As long as all dangerous actions are checked with the compliance checker, we know the security policy is being followed

Distributed/Decentralized Policy

- In a “perfect world”, the policy is in one place, specified by one person or entity
- But in the real world, different parts of the policy often come from different places
 - delegation of authorization
 - different administrators for different services
 - multiple requirements for access
- You may not even be able to look at the whole policy in one place
- Scale here means complexity & distribution

Policies and credentials do similar things

- A *policy* tells *who* is trusted to do *what*
 - *who* might be a public key
 - *what* is some potentially “dangerous” action
 - spend money, claim to be “matt blaze”, access a document
- A *credential* delegates trust to *someone else*
 - *someone else* might also be a public key (e.g., a CA)
- Distributed systems blur the line between policies and credentials
 - a credential is a policy signed by someone trusted

Public Key Infrastructure

- Why don't certificates and PKIs solve everything?
 - applications want an answer to this question:
 - “is this the correct public key for this purpose?”
 - current applications need ad hoc mechanism
 - PKI systems quietly restate this by answering another question instead:
 - “who owns this public key?”
 - X.509 certificates are good at doing this
- The two questions aren't quite the same...

Why is PKI not the solution?

- Focuses authorization on identity
 - turns a hard problem into a harder one
- Encourages outsourcing of exactly what you shouldn't outsource
 - identity management
- Creates additional points of failure
- Encourages completely artificial intermediaries who seek to fill lucrative (and unneeded) vacuum
 - certificate authorities
 - OS & browser vendors

Classic

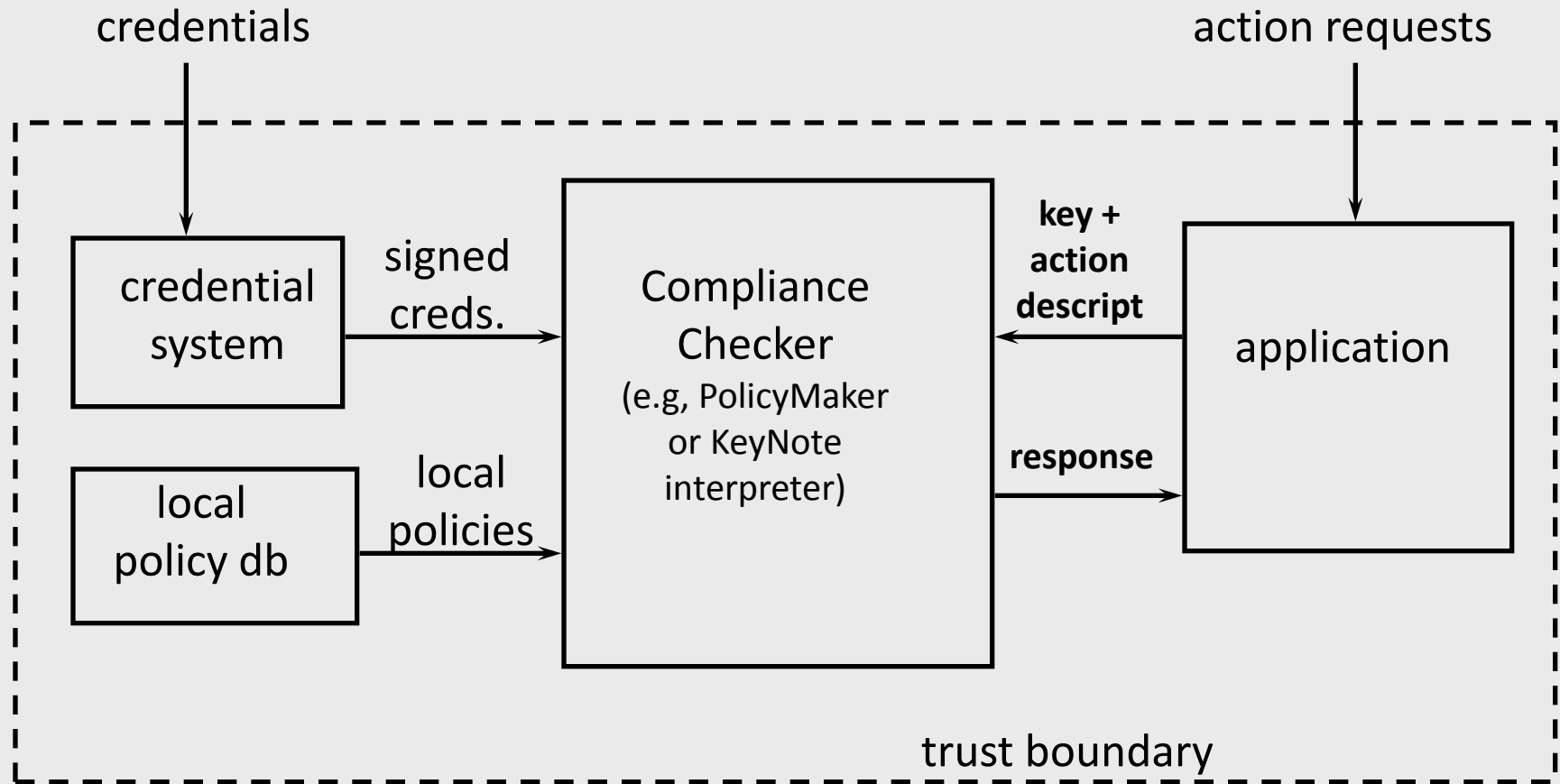
Trust Management Principles

- Separate mechanism from policy
 - application-specific data, general mechanisms
 - certificate-based systems get this backwards!
- Use a general language for writing application-specific policies and credentials
- Interpreter for this language can serve as a compliance checker that applications call to test whether an action is allowed based on policy & credentials

Classic Trust Management Elements

- A language for *Actions*
 - operations with security consequences for applications
- A naming scheme for *Principals*
 - entities that can be authorized to request actions
- A language for *Policies*
 - govern the actions that principals are authorized for
- A language for *Credentials*
 - allow principals to delegate authorization
- A *Compliance Checker* and interface
 - service that determines whether a requested action should be allowed, based on policy and a set of credentials

Classic Trust Management Architecture



Early Trust Management Languages

- PolicyMaker
 - Blaze, Feigenbaum, Lacy, 1996
 - Compliance checking semantics formalized in Blaze, Feigenbaum, Strauss, 1998
 - very general, designed more for study than use
- KeyNote
 - Blaze, Feigenbaum, Ioannidis, Keromytis 1997
 - defined in RFC 2704
 - designed to be used, especially in Internet apps
- Both share same basic semantic structure

The *KeyNote* Trust Management System

- *Actions* are represented as name/value pairs
 - Semantics of attributes are defined by application
- *Principals* can be arbitrary names or public keys
- Common language for policies and credentials
 - “Assertions” authorize a principal to perform actions that pass a predicate testing the action attributes
 - Built in delegation scheme: credentials just signed policies
 - Monotonic: adding an assertion can never cause something that was authorized to not be authorized
- KeyNote evaluates action against policies & credentials and returns advice to application

KeyNote History

- Designed in 1997-1999
 - “standardized” in RFC-2704 in 1999
- Successor to PolicyMaker (1996)
 - PolicyMaker was intended as a system to study trust management concepts and theory
 - KeyNote was intended for actual use
- Successful in that:
 - it was useful for everything we intended it for
 - it was also useful for some applications we didn't envision
- But not exactly the language we would design today

KeyNote Example

(policy and authorization cert)

```
Authorizer: "POLICY"  
Licencees: "DSA:1f203faa2babd11ffe"  
Conditions: application=="spend_money"  
            && value < 50000;
```

```
Authorizer: "DSA:1f203faa2babd11ffe"  
Licencees: "DSA:23dd11ff12efcafeff"  
Conditions: application == "spend_money"  
            && value < 10000;  
Signature: "093a3134ffa38172200333110a2bc"
```

KeyNote applications

- KeyNote was designed for small- and medium-scale internet applications
- Integrated into policy layer for
 - Apache web server
 - IPSec VPN management
- Used inside AT&T

Trust Management and Large-Scale Systems

- In the 1990's, conventional wisdom was that hierarchical certificates (e.g., X509) were as the “magic bullet” solution to trust
 - but unfortunately, PKI is hierarchical, inflexible
 - even military organizations aren't as hierarchical as X509 certificate infrastructures assume!
- We developed the original trust management model partially as a response to X.509 model
 - the real world is much less hierarchical
 - needs flexibility and decentralized control.
- Large scale government systems that require flexible controls (e.g., GIG)

Limitations of the “Classic” Trust Management Model

- Trust management layer is a powerful architectural model, but does not address:
 - enterprise infrastructure and revocation
 - policies for changing external conditions
 - e.g., behave differently when offline
 - complex quantitative decision making
 - interaction with devices/systems/entities outside the policy enforcement layer
- These are all requirements in large-scale systems

Example:

Dynamic Network Policy

- Often makes sense to have a very restrictive, hierarchical policy in normal operation
- But under crisis conditions (in the military, a war; in the civil world, a DDoS attack), it may make sense to relax the policy in specific ways
 - e.g., allow logins based on expired credentials
- Traditional security policy approaches don't do this well or securely
 - how to quantify and detect that this has happened
 - how to be sure the attacker can't artificially create the conditions that force you to relax policy

A Dynamic Trust Management Framework

- Inputs beyond policy and credentials
 - human input
 - risk-based data (e.g., output from network sensors to reliably detect changing conditions)
- More expressive languages that account for variety of input and more complex policy calculations
- Infrastructure to support policy distribution and revocation
- But all still encapsulated in a single trust management layer

^Some future directions

- Trust management at the cyber-physical interface
 - physical security systems
 - increasingly characterized by tight coupling between electronic systems and human interface – *people* are part of the system, and so are computers
 - existing systems integrate the human-computer policy engine poorly
 - Electronic voting
 - what are the trust requirements?
 - how can we quantify & manage risk?
 - what to do when irregularities are detected?